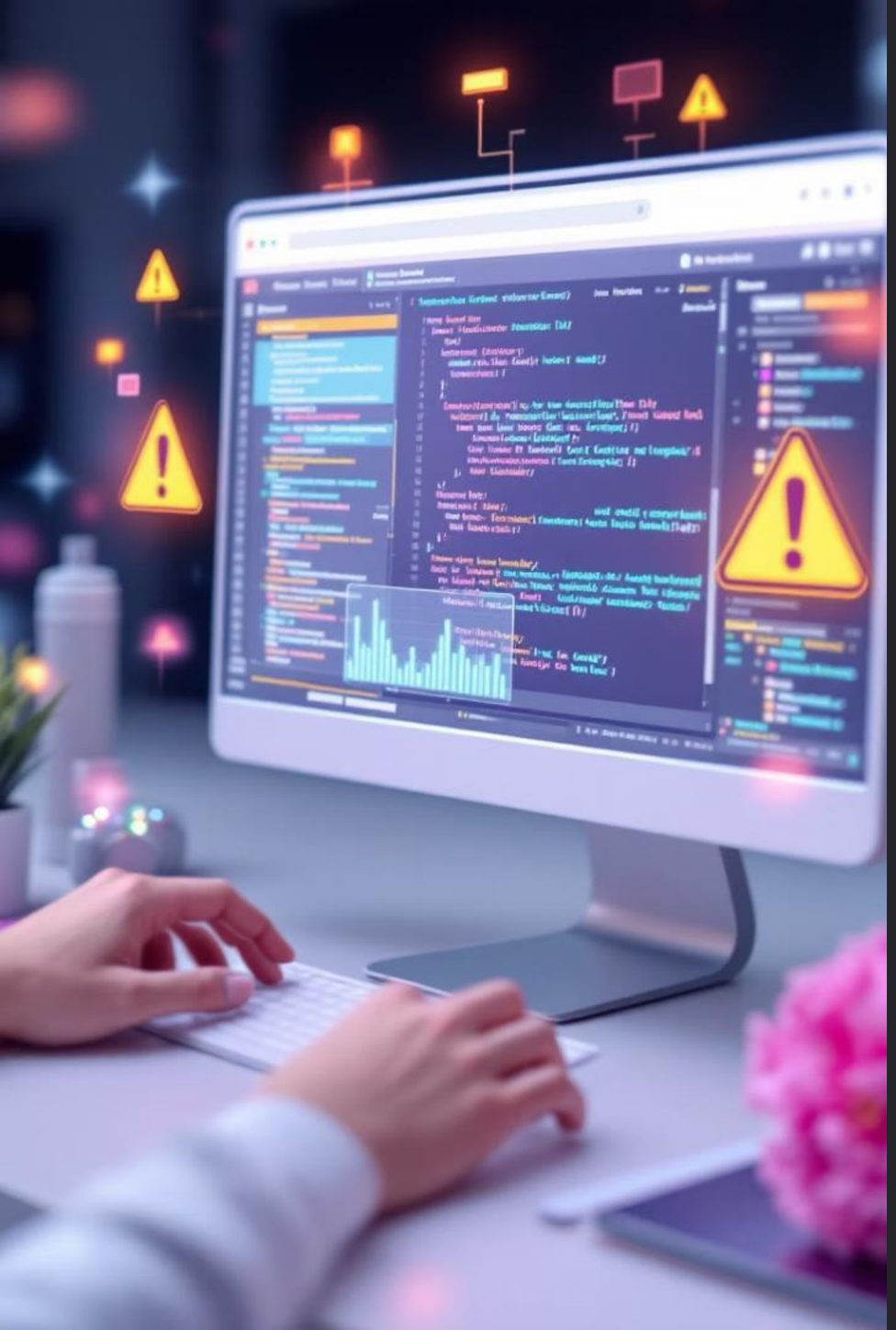
스캐너 vs 수동 테스트

보안 취약점 분석의 두 가지 접근법



스캐너란 무엇인가?

스캐너의 개념

웹페이지나 서버를 자동으로 스캔해 취약점을 찾는 보안 도구

스캐너의 역할

열린 포트, 알려진 취약점 등을 탐지하고 리포트

현실적 제약

이상적으로는 완벽하지만 실제로는 오탐, 누락 등 여러 제약 존재

스캐너의 기본 원리

대상 페이지 분석

공격할 요소들을 찾기

공격용 데이터 삽입

해당 요소들에 공격용 데이터를 삽입하여 페이지를 호출

결과 판단

호출된 결과 값을 기반으로 공격의 성공 여부를 판단

SQL 인젝션 스캐너 구현

공격 요소 파악

폼의 action, method, input type 분석

공격용 데이터 정의

예: ', ' OR '1'='1, tom

성공 여부 판단

SQL 에러 메시지 포함 여부 확인

예: pyodbc.ProgrammingError

SQL 인젝션 스캐너 구현

1. 스캐너 동작 시나리오 잡기
2. 모듈 설치하기
3. SQL 스캐너 코드 만들기
4. 스캐너 테스트

SQL 인젝션 스캐너 구현

1. 스캐너 동작 시나리오 잡기

SQL 인젝션 스캐너 구현

2. 모듈 설치

Pip install requests

기능 : 웹사이트에 요청을 보내고,HTML등의 응답을 받아오는 역할

예시: 뉴스 사이트HTML 코드 받아오기

Pip install beautifulsoup4

기능: 받아온 HTML을 사람이 읽을 수 있게 파싱(구조화)해서 필요한 정보 추출

예시: HTML에서 제목 태그만 뽑아오기

SQL 인젝션 스캐너 구현

1. 일반 클래스 vs 데이터 클래스 예제 코드 작성

일반 클래스 vs 데이터 클래스

항목	일반 클래스	데이터 클래스
__init__()	수동 작성 필요	자동 생성
__repr__()	직접 작성	자동 생성
== 비교	주소 비교 → False	값 비교 → True
코드 길이	길고 반복 많음	매우 간결함
용도	커스텀 로직 필요 시	단순한 데이터 저장 구조

@dataclass는 데이터 저장용 객체에 특화된 클래스 자동화 도구로,
코드 양을 줄이고 가독성을 높여 줍니다.

SQL 인젝션 스캐너 구현

3. SQL 스캐너 코드 만들기

SQL 인젝션 스캐너 구현

4. SQL 스캐너 확인 하기

SQL 인젝션 스캐너 구현

5. 실제 공격 시나리오 기반 인젝션 문자열 확장

- "' OR '1'='1", # 항상 참이 되도록 만드는 고전 인젝션
- "' OR 1=1--", # 주석 처리 포함
- "'; DROP TABLE users--", # 테이블 삭제 시도
- "' UNION SELECT NULL--", # 유니온 기반 우회
- "' AND 1=0--", # 참/거짓 비교 실험
- "admin' --", # 인증 우회 시도

SQL 인젝션 스캐너 구현

6. 로그 저장 기능 추가

- 대안: 결과를 실시간 출력만 하지 말고, 파일로 저장하도록 개선

XSS 인젝션 스캐너 구현

공격 요소

SQL 인젝션과 동일하게 폼 요소를 파악

공격용 데이터

alert('reflected xss run')과 같은 스크립트 코드 사용

성공 여부 판단

공격에 사용한 문자열이 HTML 엔터티 치환 없이 그대로 노출되는지 확인

XSS 인젝션 스캐너 구현

스캐너 동작 시나리오 잡기

1. 공격할 요소들은 무엇인가?
2. 공격용 데이터는 어떻게 정의하는가?
3. 성공 여부는 어떻게 알 수 있나?

XSS 인젝션 스캐너 구현

스캐너 동작 코드 작성

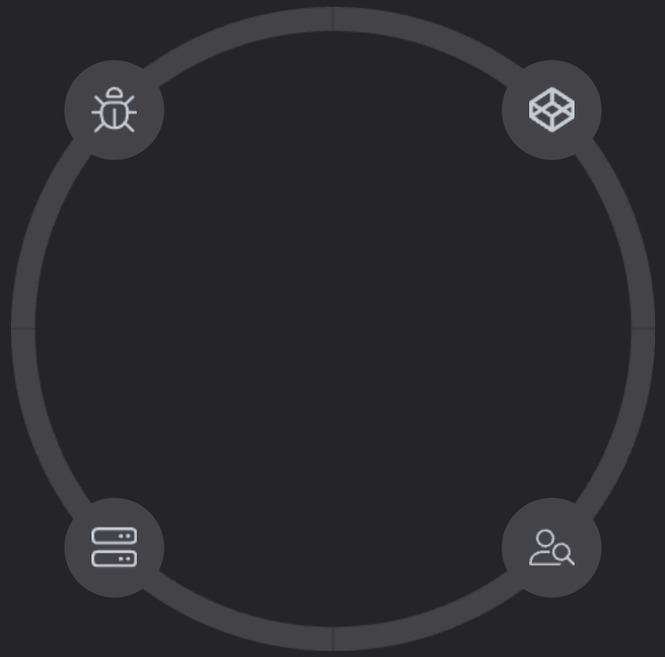
스캐너의 한계

에러 기반 판단의 한계

에러를 숨기는 설계에서는 취약점 판단이 어려움

블랙박스 관점의 제약

내부 로직을 몰라 정확한 분석이 어려움



페이지 파싱의 어려움

복잡한 구조의 웹에서 입력 요소 자동 탐지가 어려움

판단 문자열의 오탐

판단 기준 문자열이 우연히 포함되면 오탐 발생 가능

수동 테스트의 장점

복잡한 시나리오 테스트

자동화하기 어려운 파일 업로드 등 복잡한 흐름 테스트 가능

다단계 검증 가능

여러 단계로 구성된 페이지에서 단계별 취약점 점검

다양한 관점 활용

블랙박스·그레이박스·화이트박스 방식 유연하게 조합 가능

도메인 지식 활용

경험과 지식을 바탕으로 전략적으로 테스트 수행 가능

기업에서는 자동화 테스트를 선행하고,
중요한 기능은 반드시 **수동으로 보안 전문가가 직접 확인**하는 방식으로 병행

테스트 종류

테스트 유형의 구분

테스트 종류

블랙박스 테스트

내부 코드나 시스템 구조를 **전혀 모른 채**, 사용자처럼 테스트하는 방식

목표	입력 → 출력이 제대로 되는지만 확인
보안예시	로그인 화면에서 admin' OR '1'='1 입력 → 우회 가능 여부 확인
장점	사용자 관점의 테스트, 현실적인 시나리오에 강함
단점	내부 로직을 몰라서 원인 분석은 어려움

테스트 종류

화이트박스 테스트

개발자처럼 내부 소스 코드, 로직, 알고리즘 등을 **전부 알고** 테스트

목표	모든 코드 흐름을 점검, 조건문/루프 등 테스트 커버리지 확인
보안예시	소스코드에서 eval() 사용 확인 → 코드 인젝션 가능성 분석
장점	논리적 오류나 숨겨진 백도어도 탐지 가능
단점	시간 소모 큼, 전체 코드가 있어야 함

테스트 종류

그레이박스 테스트

일부 내부 정보는 알고, 일부는 모르는 상태에서 테스트

목표	외부 입력 기반 테스트 + 내부 구조의 일부 활용
보안예시	API 문서와 DB 구조 일부 알고 요청 변조 테스트
장점	효율성과 정확성의 균형
단점	외부 테스트보다 깊이 부족, 내부 테스트보단 정보 제한됨

요약 비교

구분	내부 구조 이해	주체 예시	강점
블랙박스	없음	사용자,해커	현실적 시나리오,빠름
그레이박스	일부 이해	테스터,감사자	균형있는 분석
화이트박스	완전 이해	개발자,보안팀	논리오류, 백도어 탐지 강함

자동 테스트와 수동 테스트의 조화

100%

상호 보완적 관계

두 방식은 서로의 단점을 보완하는 좋은 수단입니다.

2X

효율성 증대

자동화는 반복 작업을, 수동은 복잡한 판단이 필요한 작업을 담당하면 효율이 높아집니다.

360°

포괄적 보안

두 방식을 함께 사용하면 더 완벽한 보안 테스트가 가능합니다.